Process Control Bus Monitoring and Analysis

Field of the Invention

The present invention relates to process control bus monitoring and analysis, and in particular to an efficient and user friendly method and apparatus for such monitoring and analysis.

Background of the Invention

Many protocols are used for communication in the process control environment. Data transferred on a bus will usually be in coded form. Data is in a numerical format (i.e. decimal format) or binary format (combination of zeros and ones) or hexidecimal format (0-9 and A to F). A group of data is usually called a packet or a frame. Frames may also consist of multiple packets. A packet defines a method of arranging real world information like source address, destination address, type of packet, data, checksum etc.

In order to verify that the bus is functioning properly and devices on the bus are working as expected, it is necessary to understand each packet of data that is sent on the bus. Generally, analysis of packets is done manually by a developer during development, by an installation engineer during installation of a system, or by a person trouble shooting an installed system to help in identifying which device is not functioning properly. Each packet is observed and interpreted by looking at manuals or other related documents. The analysis is error prone, as the packet structure becomes complex for advanced control protocols.

In one protocol, Modbus, a master device queries a slave device, which gives a response to the query. A typical master packet might look like "0102000000001B9CA", while a slave packet might look like "0102020000B9B8". The data packet from a slave consists of slave address "01", function codes "02" identifying the type of information, data "00000001" and a cyclical redundancy check (CRC) "B9CA". For a master packet, the first byte is a slave address "01", followed by a function code "02", CRC "B9B8" and remaining data "020000", where "02" is data length, "0000" is data with some meaning attached. Typical meanings include whether the information is regarding the status of the

device or control data of the device. A CRC is used to check whether the packet is received correctly or not. Manually determining all of this information can be a cumbersome, error prone process.

There is a need for analysis tools to aid in interpretation of packets which is easy to use and reduces errors.

Summary of the Invention

A computer implemented method analyzes packets of data in frames on a process control bus. A frame is first selected, and a text file representative of specifications for the process control bus communication protocol is used to identify function code formats and calculate values and descriptions for the fields. A display of the values and fields is provided to the user. The display comprises a multi-pane screen of information that facilitates a user selecting a packet on the bus and viewing an interpretation of the data in the packet including values and descriptions for function codes as well as identifying the origination of the packet.

In one embodiment, the text file is converted to a data structure to facilitate matching the fields in the frames to the information from the text file. In a further embodiment, computer instructions for causing a computer to perform the method are encoded and stored on a computer readable medium.

In a further embodiment, a system for interpreting packets on a process control bus has a communication module for coupling to the process control bus. A receive queue receives a frame from the communication module. A receive module compares records in the frame with records in an interpretation file to provide a user viewable interpretation of the frame. A statistics module is coupled to the receive queue for generating statistics regarding frames received from the process control bus in a further embodiment.

Further embodiments include a data link layer, an interpretation editor for modifying the interpretation files, a log file coupled to the interpretation file, and an offline viewer coupled to the log files and interpretation file that interprets data packets in frames.

Brief Description of the Drawings

Figure 1 is a block diagram of a process control system that implements a protocol analysis method in accordance with the present invention.

Figure 2 is a flow chart showing the protocol analysis method of Figure 1.

Figure 3 is a block diagram of a protocol analyzer for implementing the analysis method of Figure 2.

Figure 4 is a computer screen shot generated by the protocol analyzer showing a frame view pane.

Figure 5 is a computer screen shot generated by the protocol analyzer showing a configure pane.

Figure 6 is a computer screen shot generated by the protocol analyzer showing a function code statistics pane.

Figure 7 is a computer screen shot generated by the protocol analyzer showing a slave address statistics pane.

Detailed Description of the Invention

In the following description, reference is made to the accompanying drawings that form a part hereof, and in which is shown by way of illustration specific embodiments in which the invention may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the invention, and it is to be understood that other embodiments may be utilized and that structural, logical and electrical changes may be made without departing from the scope of the present invention. The following description is, therefore, not to be taken in a limited sense, and the scope of the present invention is defined by the appended claims.

A process control system implementing the present invention is indicated generally at 100 in Figure 1. Figure 1 is a high level block diagram of one kind of process control system that can benefit from the present invention. It should be

recognized that many other process control systems may benefit, including distributed processing control systems. A controller 110 sends and receives communications via a bus 115 to multiple devices 120, 125 and 130. The bus is a redundant bus in some embodiments, and the devices include sensors, actuators, motors, valves, and many other types of devices commonly used in a process control environment.

A typical controller includes one or more processors 135 and memory devices 140 coupled to a longer term non-volatile storage device 145. Again, many other types of controllers may also benefit from the present invention. A display device is coupled to the controller 110 to provide visual or audible information to a person working with the system. One or more input devices 155 are also provided. These include touch screens, keyboards, pointing devices, voice recognition devices and any other means by which a person or other machine may communication with the controller 110.

Communications on bus 115 occur between the controller and devices in accordance with a predetermined bus protocol. One such protocol comprises a Modbus protocol. Each protocol has specifications describing the format and position, or data pattern of information being transferred over the bus.

In the Modbus protocol, a master device queries a slave device, which gives a response to the query. A typical master packet might look like "010200000001B9CA", while a slave packet might look like "0102020000B9B8". The data packet from a slave consists of slave address "01", function codes "02" identifying the type of information, data "00000001" and a cyclical redundancy check (CRC) "B9CA". For a master packet, the first byte is a slave address "01", followed by a function code "02", CRC "B9B8" and remaining data "020000", where "02" is data length, "0000" is data with some meaning attached. The Modbus protocol specifications provide the following information to aid in deciphering the data patterns.

#02h, Status Data Read

02h, 0000, Enthalpy status, 0, 0, U1, , 0, , {0 = Not OK to economize, 1 = OK to economize}

02h, 0000, Power exhaust status, 0, 1, U1, , 0, , {0 = Inactive, 1 = Active}

02h, 0000, Enthalpy Configuration, 0, 2, U2, , 0, , {0 = Dry bulb, 1 = Ref., 2 = Comparative}

02h, 0000, Reserved, 1, 4, U4, , 0, , ,

Enthalpy Status can be calculated manually as follows:

02h, 0000, Enthalpy status, 0, 0, U1, , 0, , {0 = Not OK to economize, 1 = OK to economize}

The above sentence has the following meaning attached to it

Field 1	Function code	02h
Field 2	Function Address	0000
Field 3	Data Name	Enthalpy status
Field 4	Start Byte	0
Field 5	Start Bit	0
Field 6	Field Type & Length	Unsigned 1 bit
Field 7	Multiplication Factor	Empty, which means 1 by default
Field 8	Offset	0
Field 9	Units	Empty, which means No units
Field 10	Interpretation Enumeration	0 = Not OK to economize, 1 = OK to economize

Enthalpy Status value = (value * multiplication factor) + offset

= (0 * 1) + 0

= 0

The Interpretation Enumeration says if enthalpy status = 0 then the user can understand it as " Not OK to Economize".

Similar calculations are done manually for Enthalpy configuration, power exhaust status and reserved.

Field Name	Value	Units & Interpretation
Enthalpy configuration	0.0	Dry Bulb configuration
Enthalpy Status	0.0	Not OK to economize
Power exhaust status	0.0	Inactive
Reserved	0	--

To automate the analysis of the data patterns, a tool that functions in accordance with the flowchart of Figure 2 is provided. A text file is used while automating this analysis. The text file contains the data, which has details about the data packets that are moving on the control bus. Values for fields such as Enthalpy, dry bulb configuration etc., are determined. When the application starts, the application reads the text file and stores the data in a data structure. Whenever the user selects a data packet for analysis, the application gets the necessary information from the data packet and looks for a matching record in the data structure. Once a match is found, the application analyzes the data and gives the information to the user interface.

The user initiates the analysis at 210, and selects a frame to analyze at 215. The function code is obtained from the frame at 220, and the function code format is searched for from the text file at 225. If the function code format is not found at 230, block 235 indicates that a new function code frame format is not available, and a new function format messages is returned at 240 to the user.

If the function code format is found at 230, a function frame format is retrieved, and frame length is initialized to equal 1 at 245. The field is then obtained at 250. Data from the frame is obtained at 255 and saved in a data structure as shown in the file shown following this paragraph. If the data is not present at 260, an error message indicating that there was insufficient data is generated at 265 and returned to the user at 270. If the data is present, a check is made for the next field at 275. If there is a next field, 1 is

added to the frame length, and the next field is obtained at 250. This loop iterates until there are no further fields, whereupon the frame length is returned at 285.

Function Code Formats:

```
#define is a programming language convention.
**************************************************
#define Fnc01Query      "Slave_Address1, FncCode1, AddHi1, AddLo1,
PointsHi1, PointsLo1, Crc2,"
#define Fnc01Response "Slave_Address1, FncCode1, ByteCount1, Data1,
Crc2,"

**************************************************

#define Fnc01Query      "Slave_Address1, FncCode1, AddHi1, AddLo1,
PointsHi1, PointsLo1, Crc2,"
Here Fnc01Query means function code 01 and message is of type Query
Slave_Address1     means field name is Slave Address and length of the
field is 1 Byte
FncCode1           means field name is Function Code and length is 1
Byte
AddHi1             means field name is address High and length is 1 Byte
AddLo1             means field name is address Low and length is 1 Byte
PointsHi1          means field name is points High and length is 1 Byte
PointsLo1          means field name is points low and length is 1 Byte
Crc2               means field name is CRC and length is 2 Bytes


#define Fnc01Response "Slave_Address1, FncCode1, ByteCount1, Data1,
Crc2," Fnc01Response means function code 01 and message is of type
Response

Slave_Address1     means Slave Address and length is 1 Byte
FncCode1           means Function code and length is 1 Byte
ByteCount1         means Byte count and length is 1 Byte
Data1              means Data and length is 1 byte
Crc2               means CRC and length is 2 bytes
```

The text file is flexible enough for the end user to modify accordingly to suit their needs. It is used by finding a value in the frame and matching it to a corresponding verbal description from the text file to determine the state of function codes. The following is a sample text file:

Example text file

; Enviracom Monitoring Tool (EMT) Message Interpretation File, Rev 0.9.3 (Working) 28-Nov-2000

#02h, Status Data Read

02h, 0000, Enthalpy status, 0, 0, U1, , 0, , {0 = Not OK to economize, 1 = OK to economize}

02h, 0000, Power exhaust status, 0, 1, U1, , 0, , {0 = Inactive, 1 = Active}

02h, 0000, Enthalpy Configuration, 0, 2, U2, , 0, , {0 = Dry bulb, 1 = Ref., 2 = Comparative}

02h, 0000, Reserved, 1, 4, U4, , 0, , ,

#01h, Control data Read,

01h, 0000, Call for Cooling, 0, 0, U1, , 0, ,{0 = No Cooling, 1 = Cooling demand}

01h, 0000, Fan state, 0, 1, U1, , 0, ,{0 = Inactive, 1 = Active}

01h, 0000, Occupancy state, 0, 2, U1, , 0, ,{0 = Unoccupied, 1 = Occupied}

01h, 0000, Enthalpy Control, 0, 3, U1, , 0, ,{0 = Disabled, 1 = Enabled}

01h, 0000, Enthalpy Mode, 0, 4, U1, , 0, ,{0 = Normal, 1 = Override}

01h, 0000, Shutdown, 0, 5, U1, , 0, ,{0 = Normal, 1 = Shutdown}

01h, 0000, Baud rate, 0, 6,U2, , 0, ,{0 = 1200, 1 = 2400, 2 = 9600, 3 = 19.2k}

01h, 0000, Slave address, 1, 0, U2, , 0, , ,

01h, 0000, Reserved, 1, 3, U6, , 0, , ,

The manual document has much more detail than is needed. It is likely that a user may have to make notes about things like what function codes are available and how they can be evaluated. The text file that is used in the present invention contains all the information needed for computing the values of fields in a frame in an easier manner. A larger sample text file follows:

; *#*#*#*#*#* Message Header Entry syntax *#*#*#*#*#*

;Frame structure is as follows

;for Query frames with function codes01, 02, 03, 04, 05, 06, 10,

;slaveid, function code, Address High, Address Low, # of Points High, # of Points Low,

;Data high, Data Low, CRC Low, CRC high

;for frames with function code 08

```
;slaveid, function code, subfunctionHi, subfunctionLo, Data Hi,
;Understanding Query frame
;Each item occupies one byte
;for Response frame
;slave id, function code, Byte count, Data High, Data Low, ..., , CRC Low, CRC high
; Slave id is not considered
;#function code
;       # is compulsory before function code
;Byte count is an integer that represents number of bytes of data present in the frame
;Data high, data low ...,  each represent one byte of data
;CRC occupies the last 2 bytes


; *$*$*$*$*$* Data Field Entry syntax *$*$*$*$*$*
;
; Function code, Data Name, Start byte, Start bit, Field Type & Length, Multiplication
;Factor, Units,[{Interpretation Enumerations}]
; ** explanation **
; Field 1 : Function code
;       Function code
; Field 2 : Data Name
;           - The text entered here will be shown as is while interpreting this field as data
;name.
; Field 3 : Start Byte
;           - The byte at which the data is to be taken for interpretation.
;           - 0 based.
; Field 4 : Start Bit
;           - The bit from which the data is to be taken for interpretation.
;           - 0 based.
; Field 5 : Field Type & Length
;           - This field gives information about the type of the field as
```

```
;                U - Unsigned
;                S - Signed
;                Along with this field type the data Length and Fraction part are also
;required to be mentioned as
;                S8.2 - Signed, 8 bits data in which 2 bit fraction.
;                U6.0/U6 - Unsigned, 6 bits data and no fraction
;            - This field is case insensitive.
; Field 6 : Multiplication Factor
;            - The data of this field is multiplied with the number given here before
;interpreting. This can be float also.
;            - The blank entry here is taken as 1.
; Field 7 : Offset
;                The data of this field is added with the calculated value
; Field 8 : Units
;            - This is field to enter Engineering Units. The Text here is shown as is in place
;of Units.
;        -sometimes % can also be associated with the Engineering Units
; Field 9 : {Interpretation Enumerations}
;            - This field will be useful to indicate text replacements when the read field data
;matches some pre-defined value.
;            - Ex :- {10-20 = Initial, 20h-30h = Final, FFh = UNKNOWN}
;                If the read value of this field falls in between Decimal 10 to 20 then
;"Initial" is shown in place of value.
;                If the read value of this field falls in between Hexa Decimal 20h to 30h
;then "Final" text is shown in place of value.
;                If the read value for this field matches Hexa Decimal FFh then
;"UNKNOWN" is shown in place of value.
;            - This entry is optional.
;            - The entries for this field has to be covered under flower braces '{'}'
;            - The entries are comma delimited. See example above.
```

; - It is allowed to make multiple entries after comma delimition. See example
;above.
;
; *$*$*$*$*$* End of Data Field Entry syntax Info *$*$*$*$*$*

#01h, Control data Read,

01h, 0000, Call for Cooling, 0, 0, U1, , 0, ,{0 = No Cooling, 1 = Cooling demand}

01h, 0000, Fan state, 0, 1, U1, , 0, ,{0 = Inactive, 1 = Active}

01h, 0000, Occupancy state, 0, 2, U1, , 0, ,{0 = Unoccupied, 1 = Occupied}

01h, 0000, Enthalpy Control, 0, 3, U1, , 0, ,{0 = Disabled, 1 = Enabled}

01h, 0000, Enthalpy Mode, 0, 4, U1, , 0, ,{0 = Normal, 1 = Override}

01h, 0000, Shutdown, 0, 5, U1, , 0, ,{0 = Normal, 1 = Shutdown}

01h, 0000, Baud rate, 0, 6,U2, , 0, ,{0 = 1200, 1 = 2400, 2 = 9600, 3 = 19.2k}

01h, 0000, Slave address, 1, 0, U2, , 0, , ,

01h, 0000, Reserved, 1, 3, U6, , 0, , ,

#05h, Control data Write

05h, 0000, Call for Cooling, 0, 0, U1, , 0, ,{0 = No Cooling, 1 = Cooling demand}

05h, 0000, Fan state, 0, 1, U1, , 0, ,{0 = Inactive, 1 = Active}

05h, 0000, Occupancy state, 0, 2, U1, , 0, ,{0 = Unoccupied, 1 = Occupied}

05h, 0000, Enthalpy Control, 0, 3, U1, , 0, ,{0 = Disabled, 1 = Enabled}

05h, 0000, Enthalpy Mode, 0, 4, U1, , 0, ,{0 = Normal, 1 = Override}

05h, 0000, Shutdown, 0, 5, U1, , 0, ,{0 = Normal, 1 = Shutdown}

05h, 0000, Baud rate, 0, 6,U2, , 0, ,{0 = 1200, 1 = 2400, 2 = 9600, 3 = 19.2k}

05h, 0000, Slave address, 1, 0, U2, , 0, , ,

05h, 0000, Reserved, 1, 3, U6, , 0, , ,

#02h, Status Data Read

02h, 0000, Enthalpy status, 0, 0, U1, , 0, , {0 = Not OK to economize, 1 = OK to economize}

02h, 0000, Power exhaust status, 0, 1, U1, , 0, , {0 = Inactive, 1 = Active}

02h, 0000, Enthalpy Configuration, 0, 2, U2, , 0, , {0 = Dry bulb, 1 = Ref., 2 = Comparative}

02h, 0000, Reserved, 1, 4, U4, , 0, , ,

#03h, Parameter Data Read/Write

03h, 40000, Maximum DMP, 1, 0, U8, 1, 0, %, ,

03h, 40000, PExPoint, 0, 0, U8, 1, 0, %, ,

; 03h, 40001, Configured DMP, 1, 0, U8, 1, 0, %, ,

03h, 40001, Configured DMP, 1, 7, U1, 1, 0, %, {0 = Configured locally (potentiometer), 1 = configured remotely(master)}

03h, 40001, Active DMP, 0, 0, U8, 1, 0, %, ,

03h, 40002, Enthalpy Setpoint, 1, 6, U2, 2, 0, bits, {0 = A, 1 = B, 2 = C, 3 = D}

03h, 40002, Current Damper Position, 1, 0, U6, 1, 0, %, ,

03h, 40002, MA SetPoint, 0, 0, U8, 0.1, 0, degrees F,

; offset exists for this 40 F

03h, 40003, IAQ Setpoint, 1, 0, U8, 10, 0, ppm, ,

03h, 40003, OAQ set point, 0, 0, U8, 10, 0, ppm, ,

#06h, Parameter Data Read/Write

06h, 40000, Maximum DMP, 1, 0, U8, 1, 0, %, ,

06h, 40000, PExPoint, 0, 0, U8, 1, 0, %, ,

; 06h, 40001, Configured DMP, 1, 0, U8, 1, 0, %, ,

06h, 40001, Configured DMP, 1, 7, U1, 1, 0, %, {0 = Configured locally (potentiometer), 1 = configured remotely(master)}

06h, 40001, Active DMP, 0, 0, U8, 1, 0, %, ,

06h, 40002, Enthalpy Setpoint, 1, 6, U2, 2, 0, bits, {0 = A, 1 = B, 2 = C, 3 = D}

06h, 40002, Current Damper Position, 1, 0, U6, 1, 0, %, ,

06h, 40002, MA SetPoint, 0, 0, U8, 0.1, 0, degrees F,

; offset exists for this 40 F

06h, 40003, IAQ Setpoint, 1, 0, U8, 10, 0, ppm,

06h, 40003, OAQ set point, 0, 0, U8, 10, 0, ppm,

;extended parameter data

06h, 40010, Extended Indoor Air Quality level, 1, 0, U8, 10, 0, ppm, ,

06h, 40010, Extended Outdoor Air Quality level, 0, 0, U8, 10, 0, ppm, ,

06h, 40011, Extended Return Air Humidity value, 1, 0, U8, 1, 0, %RH, ,

06h, 40011, Extended Outdoor Air Humidity value, 0, 0, U8, 1, 0, %RH, ,

06h, 40012, Extended Return Air Temperature vale, 0, 0, U16,0.1, 0, degrees F,

,

06h, 40013, Extended Outdoor air temperature value, 0, 0, U16, 1, 0, degrees F,

,

#10h, Parameter Data Read/Write

10h, 40000, Maximum DMP, 1, 0, U8, 1, 0, %, ,

10h, 40000, PExPoint, 0, 0, U8, 1, 0, %, ,

10h, 40001, Configured DMP, 1, 0, U8, 1, 0, %, ,

10h, 40001, Configured DMP, 1, 7, U1, 1, 0, %, {0 = Configured locally

(potentiometer), 1 = configured remotely(master)}

10h, 40001, Active DMP, 0, 0, U8, 1, 0, %, ,

10h, 40002, Enthalpy Setpoint, 1, 6, U2, 2, 0, bits,{0 = A, 1 = B, 2 = C, 3 = D}

10h, 40002, Current Damper Position, 1, 0, U6, 1, 0, %, ,

10h, 40002, MA SetPoint, 0, 0, U8, 0.1, 0, degrees F,

; offset exists for this 40 F

10h, 40003, IAQ Setpoint, 1, 0, U8, 10, 0, ppm,

10h, 40003, OAQ set point, 0, 0, U8, 10, 0, ppm,

;extended parameter data

10h, 40010, Extended Indoor Air Quality level, 1, 0, U8, 10, 0, ppm, ,

10h, 40010, Extended Outdoor Air Quality level, 0, 0, U8, 10, 0, ppm, ,

10h, 40011, Extended Return Air Humidity value, 1, 0, U8, 1, 0, %RH, ,

10h, 40011, Extended Outdoor Air Humidity value, 0, 0, U8, 1, 0, %RH, ,

10h, 40012, Extended Return Air Temperature vale, 0, 0, U16,0.1, 0, degrees F,

,

10h, 40013, Extended Outdoor air temperature value, 0, 0, U16, 1, 0, degrees F,

,

#04h, Variable Data Read

04h, 0000, Indoor Air Quality Level, 1, 0, U8, 10, 0, ppm, ,

04h, 0000, Outdoor Air Quality Level, 0, 0, U8, 10, 0, ppm, ,

04h, 0001, Mixed Air Temperature Sensor Value, 1, 0, U8, 0.1, 0, degrees F, ,

04h, 0001, Return Air Temperature Sensor Value, 0, 0, U8, 0.1, 0, degrees F, ,

04h, 0002, Return Air Humidity Sensor Value, 1, 0, U8, 1, 0, %Rh, ,

04h, 0002, Outdoor Air Humidity Sensor Value, 0, 0, U8, 1, 0, %RH, ,

04h, 0003, Outdoor Air Temperature Sensor Value, 0, 0, U16, 1, 0, degrees F

#08h, Diagnostics

08h, ,0, 0, U1, , 0, ,{1 = Acutator fault}

08h, , ,0, 1, U1, , 0, ,{1 = outdoor air quality sensor Fault}

08h, , ,0, 2, U1, , 0, ,{1 = Return Air Humidity Sensor Fault}

08h, , ,0, 3, U1, , 0, ,{1 = Return Air Temperature Sensor Fault}

08h, , ,0, 4, U1, , 0, ,{1 = Outdoor Air Quality Sensor Fault}

08h, , ,0, 5, U1, , 0, ,{1 = Outdoor Air Humidity Sensor Fault}

08h, , ,0, 6, U1, , 0, ,{1 = Outdoor Air Temperature Sensor Fault}

08h, , ,0, 7, U1, , 0, ,{1 = Mixed Air Temperature Sensor Fault}

08h, , ,1, 0, U1, , 0, ,{1 = RAM Write/Readback fault}

08h, , ,1, 1, U1, , 0, ,{1 =ROM Checksum fault}

08h, , ,1, 2, U1, , 0, ,{1 = EEPROM Checksum fault}

08h, , ,1, 3, U1, , 0, ,{1 = Thermostat fault}

; 08h, , ,1, 4, U4, , 0, ,{1 = Reserved }

; interpretation or what to do if one of above errors occurs

08h, , ,0 ,0 , U16, , 0, ,{00 = Echo, 01 = Reinitialization, 02 = Return Diagnostics Register, 04 = Set Listen Only Mode, 0A = Clear Counters, 0B = Bus Message Count, 0C = Bus Communication, OD = Bus exception Count, 0E = Slave Message Count, 0F = Slave no response count}

#11h, Slave Configuration Data

11h, , Slave Type, 0, 0, U8, , 0, , ,

11h, , Run Indicator, 1, 0, U8, , 0, , ,

11h, , Version, 2, 0, U8, , 0, , ,

11h, , Revision, 3, 0, U8, , 0, , ,

#13h, Reset

A protocol analyzer is shown generally at 300 in Figure 3. A process control bus 310 is the communication network through which devices 312, 314 and 316 communicate with each other. A communication module 320 deals with monitoring (reading data from the bus 310 and writing data on to the bus). A network data link layer 325 has functionality to identify the packets of data. A transmit queue 330 and transmit module 335 deal with transmitting the data of user interest.

A scripting module 340, master node simulation module 345 and slave node simulation module 350 are used for device simulation and help in testing the process control system. A receive queue 353 is used to receive the data that is moving on the process control bus 310. A receive module 355 and statistics module 360 are used to display the received data packets according to the protocol specifications. Receive module 355 interacts with an interpretation file 365 to gives interpretation for the frames that are moving on the bus.

Interpretation file 365 is used to analyze the data packets, and an interpretation editor 370 is used to modify this interpretation file 365. An offlineviewer 375 also uses the interpretation file 365 for interpreting the data packets. Offlineviewer 375 uses a log file 380 that contains the data that is received on the process control bus. The log file 380 is generated by the receive module 355.

Graphical screens depicting the analysis and interpretation are shown in Figures 4-7. A number of tabs are provided in each of the screens to provide for ease of use. Figure 4 shows a frame view screen 410 associated with a frame view tab 415. Other tabs include a configure tab 420, a function code statistics tab 425 and a slave address statistics tab 430. The frame view screen provides the user an easy way to view the data that is moving on the bus. This view has various panes, which helps user in viewing and understanding the data that is moving on the bus.

A scroll bar 435 provides for ease of navigation between frames. The user can view multiple frames in a pane 440. Before displaying the data packet to the user the data is split in the standard format. The user can select any message that is displayed in pane 440 and view the detailed interpretation of the same in panes 445, 450 and 455.

Pane 445 displays information regarding whether the data packet is sent by master or slave and slave address and the function code of it. Pane 450 provides a view of dissection of the data packet to the next level. Pane 445 provides a display of the interpretation of all the fields and their numerical values and their Engineering units and interpretation if any obtained from the text file. For example, the enthalpy configuration has a value of 0.0, which corresponds to a dry bulb configuration. This information is conveniently provided by comparing values in a frame or packet to the text file. Other information presented in pane 455 describes the enthalph status and not OK to economize and the power exhaust status as inactive. Engineering units (Engg Units) are the units in which a particular value is measured. For example, temperature can be measured in Celsius and Fahrenheit. Thus, the value of the data and the units are required for the information to be meaningful.

The configure tab 420 provides access to a configuration screen 510 in Figure 5. It provides information to help the user in configuring and setting options for monitoring the control bus. Information, such as the communications port and baud rate are provided. Configuration of logs is also provided.

The function code statistics tab 425 provides access to a function code statistics screen 610 in Figure 6. The slave address statistics tab 430 provides access to a slave address statistics screen 710 in Figure 7. The function code statistics screen 610 and lave

address statistics screen 710 provide statistical information to users, such as function code, frame number, error number, master number and slave number.

The interpretation is customizable by the end user accordingly whenever the need for it arises. The whole interpretation is stored in the text file, which the user can modify. In that way, the user has both customization to their requirement and transparency.

Conclusion

A computer implemented method analyzes packets of data in frames on a process control bus. A frame is first selected, and a text file representative of specifications for the process control bus communication protocol is used to identify function code formats and calculate values and descriptions for the fields. A display of the values and fields is provided to the user. The display comprises a multi-pane screen of information that facilitates a user selecting a packet on the bus and viewing an interpretation of the data in the packet including values and descriptions for function codes as well as identifying the origination of the packet.

The automated display of information regarding packets on a bus saves considerable time and effort in installation and troubleshooting operation of the bus. It is also much more accurate than prior manual processes. This application is intended to cover any adaptations or variations of the present invention. It is manifestly intended that this invention be limited only by the claims and equivalents thereof.